

# A Self-organized Clustering-based Overlay Network for Application Level Multicast

Khaled Ragab\*

King Faisal University, College of Computer Sciences and Information Technology, Saudi Arabia  
kabdultawab@kfu.edu.sa

Akinori Yonezawa

Tokyo University, Computer Science Department, Japan.  
yonezawa@yl.is.s.u-tokyo.ac.jp

**Abstract**— Peer-to-Peer Overlay networks enabling an end-end application level-multicast service have drawn enormous attention. This paper proposes a self-organized  $\langle \delta, \mu \rangle$ -Multicast Overlay Network ( $\langle \delta, \mu \rangle$ -MON) that enables an efficient end-end application level multicast. The  $\langle \delta, \mu \rangle$ -MON is organized into MON-Clusters, where  $\delta$  is the maximum physical number of hops between any two end-nodes in each MON-Cluster. Thus, end-users accessing from different areas are able to broadcast data into each MON-Cluster within a constant  $\delta$  physical number of hops. However in a dense-mode, increasing the size of MON-Clusters induces a long logical diameter that manifests high stress per physical links and high delay to disseminate data. Thus, this paper limits the expansion of each MON-Cluster within a specific logical diameter  $\mu$ . Each MON-Cluster is constructed with at most  $\mu$  logical hops over  $\delta$  physical hops. Thus, end-users are able to efficiently disseminate data into each MON-Cluster within  $\mu$  logical hops over  $\delta$  physical hops with reasonable stress per physical links and bandwidth consumption.

**Index Terms**— Network Services and Applications, Application level-multicast, Peer-to-Peer Overlay Network.

## I. INTRODUCTION

THE Peer-to-Peer (P2P) overlay networks are decentralized systems in nature, without any hierarchical organization or centralized control. P2P overlay systems go beyond services offered by client-server systems by having symmetry in roles where a client may also be a server. As a result, P2P overlay networks have drawn increasing attention nowadays and have been widely deployed on the Internet for various purposes, including distributed data storage [1][2], file sharing networks [3], information retrieval [4] and information dissemination [5]. We can view P2P overlay network models spanning a wide spectrum of the communication framework that specifies a fully-distributed, cooperative network design with peers building a self-organizing system. Depending on the sequence of construction of the control and data topologies, the P2P overlay networks are classified into three different approaches: *tree-first* (e.g. *ALMI* [6], *Yoid* [7]), *mesh-first* (e.g. *Narada* [8], *Scattercast* [9]) and

*implicit* (e.g. *CAN* [10], *Community Overlay Network CON* [11]) approaches [12].

This paper proposes an implicit self-organized overlay network model that enables scalable and efficient information dissemination. It introduces an autonomic  $\langle \delta, \mu \rangle$ -Multicast Overlay Network ( $\langle \delta, \mu \rangle$ -MON or simply *MON*) construction scheme for efficient end-end multicast service. Moreover, this paper looks into the question of how the deployment of such *MON* might be utilized to support multicast services and applications. We outline the design of an application-level multicast scheme built upon *MON*. The *MON* is organized into *MON-Clusters*.

The physical distance between any two end-nodes in each *MON-Cluster* is less than or equal  $\delta$  physical hops. In large-scale and dense networks, increasing the size of *MON-Cluster* yields a long logical diameter that demonstrates high stress per physical links and high delay to disseminate data. Thus, this paper designates the logical diameter,  $D_{\tau}^{(j)}$  of each *MON-Cluster*  $j$  at instance of time  $\tau$  is bounded by  $\mu$ . The *MON-Cluster* is not allowed to expand than  $\mu$  logical hops over  $\delta$  physical hops. This is called  $\langle \delta, \mu \rangle$ -Clustering property. We argue the efficiency of the multicast service over the proposed *MON* to the  $\langle \delta, \mu \rangle$ -Clustering property, in other words, to the topology awareness construction of each *MON-Cluster* over  $\delta$  physical hops and to the expand limitation diameter  $\mu$ . That limits the expansion of the *MON-Cluster*'s diameter. Thus a message can be broadcasted over *MON-Cluster* within at most  $\delta, \mu$  number of physical hops and number of logical hops respectively. The remainder of this paper is organized as follows. Section 2 discusses some related works. Section 3 exhibits 2-tier architecture of the  $\langle \delta, \mu \rangle$ -MON, construction scheme and fault-tolerance. Section 4 shows the simulation setup and performance results. Finally, section 5 draws our conclusion.

## II. RELATED WORKS

This section discusses some related works of overlay construction for Application level multicast. In previous

\*On leave College of Science, Ain Shams University, Cairo, Egypt.

work, several research proposals have argued for *application-level* multicast [7, 8, 9] as a more tractable alternative to a network-level multicast service. They have described designs for such a service and its applications. The majority of these proposed solutions (for example [8, 9]) typically involve having the members of a multicast group self-organize into an essentially random application-level mesh topology. Over which a traditional multicast routing algorithm such as DVMRP [13] is used to construct distribution trees rooted at each possible traffic source. Such routing algorithms require every node to maintain state for every other node in the topology. If the multicast group is composed of  $N$  nodes then each node should maintain state for  $N-1$  nodes. Hence, although these proposed solutions are well suited to their targeted applications. Their use of a global routing algorithm limits their ability to scale to large (many thousands of nodes) group sizes. Moreover, it limits their ability to operate under conditions of dynamic group membership. Bayeux [14] is an application-level multicast scheme. It scales to large group sizes but restricts the multicast service model to a single source. In contrast to the above schemes, CAN-based multicast [23] and  $\langle \delta, \mu \rangle$ -Multicast Overlay Network can scale to large group sizes without restricting the service model to a single source. CAN overlay network employs *landmark-based* placement at join time for underlying network topology awareness. At high density network, every CAN node has increasingly many close-by nodes that manifests high stress per physical links and long delay for multicasting data. A CAN network with  $N$  nodes requires at most  $dN^{1/d}$  logical hops to multicast a message [23], where each node maintains  $2d$  neighbors. However, if this network is organized into small clusters with size  $S$  where  $S \ll N$  then a message can be broadcasted in parallel over each cluster within small number of logical hops. In a high dense network, if CAN's overlay network is organized into clusters while preserving the underlying network topology awareness property. Then the cluster diameter increases and manifests high stress per physical links and long delay for multicasting data. To overcome this problem, it is required to limit the clusters' diameter. For that reason, this paper organizes the  $\langle \delta, \mu \rangle$ -Multicast Overlay Network into *MON-Clusters* that are not only aware of the underlying topology but also limit the clusters' diameter. To our knowledge, there is no article that constructs the overlay network based on  $\langle \delta, \mu \rangle$ -Clustering property for efficient multicast service. However, for efficient routing and tracking some centralized and distributed approaches have been studied. These approaches enable the construction of sparse and locality preserving representations for arbitrary networks [15]. They used algorithms that divide a given static network  $G$  into small connected partitions with least possible number of nodes by means of a synchronous centralized/ distributed computation on  $G$ . These approaches were not capable of being applied to divide a

dense and dynamic overlay network into clusters under the situations of frequent peers join and leave. In contrast, this paper focuses on step-step construction of the dynamic overlay network with maintaining the  $\langle \delta, \mu \rangle$ -Clustering property.

The proposed  $\langle \delta, \mu \rangle$ -MON is a new generation of P2P overlay network. It aims to provide high resilience through  $\langle \delta, \mu \rangle$ -Clustering techniques, clusters as *regular-graph* composed of *Hamilton* cycles. Thus, the proposed  $\langle \delta, \mu \rangle$ -MON step-step construction scheme enables an efficient multicast within each cluster over  $\delta$ , a constant physical hops, and  $\mu$  constant logical hops at most. MON is not only aware by the underlying network topology but also aware with multicast cluster size. We argue that to the  $\langle \delta, \mu \rangle$ -MON clustering technique that organizes MON into limited size clusters of nearby nodes in the underlying network. Thus, the multicast delay and network load are reasonable as shown in figures 4, and 5.

### III. $\langle \delta, \mu \rangle$ -MULTICAST OVERLAY NETWORK

This section describes the proposed multicast overlay network architecture and construction algorithm.

**Definition 1 ( $\langle \delta, \mu \rangle$ -Clustering):** Given a graph  $G = (V, E, \langle \delta, \mu \rangle)$ , where vertex  $v \in V$ , edge  $e \in E$ . The pair  $\langle \delta, \mu \rangle$  is integer. It clusters the graph  $G$  into connected sub-graphs (*MON-Cluster*)  $C$ . Each cluster has a diameter less than or equal the upper bound diameter  $\mu$  over  $\delta$  physical links. In other words, the distance between any two vertices in the *MON-Cluster* is less than  $\langle \delta, \mu \rangle$ , where  $\delta, \mu$  are the *clustering parameters*.

#### A. $\langle \delta, \mu \rangle$ -MON Architecture

The 2-tiers architecture of MON is shown in figure 1 and explained as follows. It is composed of multiple *MON-Clusters*  $C_k$ ;  $k=1, \dots, \omega$  in level 1. Each  $C_k$  has a *tutor* node called  $t_k$ . The physical distance between any  $i \in C_k$  and  $t_k$  is  $d(i, t_k) \leq r$ ;  $\delta=2r$  as shown in figure 1. The *distance* between two vertices  $i$  and  $j$ ,  $h(i, j)$ , is the number of logical hops of a shortest path between  $i$  and  $j$ . Its maximum value over all pair of vertices,  $D_\tau^{(k)} = \max \{h(i, j); \forall i, j \in C_k\}$ , is the *diameter* of the *MON-Cluster*  $k$  at instance of time  $\tau$ . Where  $D_\tau^{(k)}$  is less than or equal the upper bound diameter  $\mu$ . In level 2 a cluster that contains all tutor nodes is constructed and called *Tutors-Cluster* as shown in figure 1. The size of the *Tutors-Cluster* is denoted by  $\omega$  and equaled to the number of *MON-Clusters* in the level one.

#### B. *MON-Cluster*

**Definition 2 (*MON-Cluster*):** The  $\langle \delta, \mu \rangle$ -MON cluster is defined by a subset of nodes which are mutually reachable by a path of length at most  $\delta$  physical hops and  $\mu$  logical hops for some fixed  $\delta$  and  $\mu$ .

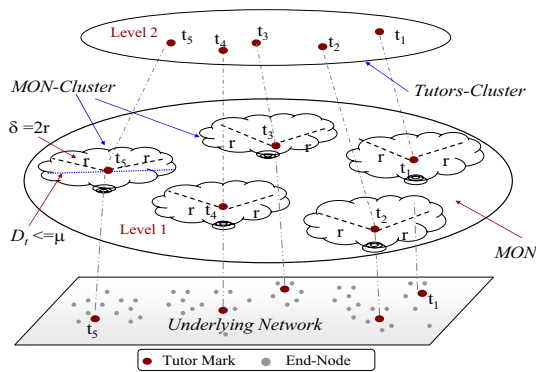


Figure 1: The 2-tiers  $\langle \delta, \mu \rangle$ -MON architecture.

**Definition 3** The size,  $|C_j|$  of a MON-Cluster  $C$  is the number of nodes in  $C$ .

Each MON-Cluster ( $C$ ) and Tutors-Cluster are constructed as a  $2\chi$ -regular graph composed of  $\chi$  independent edge-disjoint Hamilton Cycles (HC) [16]. Each node has  $2\chi$  neighbors (node connectivity). Those neighbors are labeled as  $\mathbf{n}_p^{(1)}, \mathbf{n}_s^{(1)}, \mathbf{n}_p^{(2)}, \mathbf{n}_s^{(2)}, \dots, \mathbf{n}_p^{(\chi)}, \mathbf{n}_s^{(\chi)}$ . For each  $i$ ,  $\mathbf{n}_p^{(i)}$  denotes the neighbor node's predecessor and  $\mathbf{n}_s^{(i)}$  denotes the neighbor node's successor on the  $i$ -th HC. For example, figure 2 shows a MON-Cluster consists of two Hamilton cycles with diameter  $D_\tau = 3$  (logical hops) over the underlying network. The physical distance between the tutor node  $t_1$  and any node is less than or equal two (i.e. radius  $r = 2$  and diameter  $\delta = 2r$ ) backbone physical hops between their routers. This paper constructed  $C$  as regular-graph for three arguments as follows. First, regular graphs are chosen because it is required that all nodes having the same degree for load balance. Second,  $C$  is constructed as an intermediate of a completely ordered regular network and a fully random network for achieving two interesting features: short network diameter and high clustering. High clustering feature is recognized if there is a high density of connections between nearby nodes, which is a characteristic of the regular topologies. Finally, the cluster  $C$  is composed of HC that enjoys the advantage of joining or leaving processes that require only local changes.

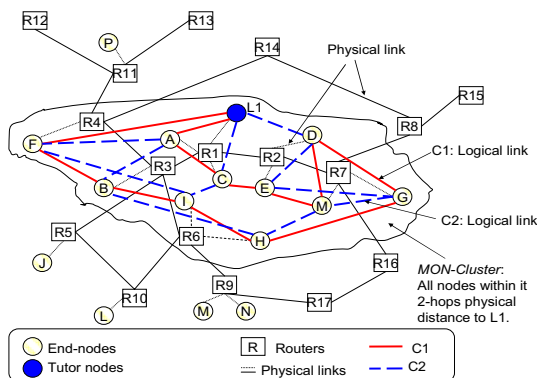


Figure 2: An example of MON-Cluster.

**Theorem 1 (Cluster-connected):**

A  $\langle \delta, \mu \rangle$ -MON graph is cluster-connected if it satisfies the following conditions:

- 1) The union of the MON-Clusters covers the whole  $\langle \delta, \mu \rangle$ -MON.
- 2) Tutors-Cluster is connected graph
- 3) For a MON-Cluster  $C_j$ , where  $|C_j| \geq 5$  there is a path from each node to every other node in the MON-Cluster.

**Proof:**

The first condition can be proved by the step-step construction scheme in section 3.3. Any new node joins the  $\langle \delta, \mu \rangle$ -MON may join an existing MON-Cluster or construct its own MON-Cluster. Thus all nodes join the  $\langle \delta, \mu \rangle$ -MON must be participating in a MON-Cluster. Thus the union of MON-Clusters covers the whole  $\langle \delta, \mu \rangle$ -MON.

The second and third conditions are satisfied because of the architecture of the MON-Cluster as a  $2\chi$ -regular graph that is composed of  $\chi$  independent edge-disjoint HC. If a Hamilton path connects every two nodes of a MON-Cluster then it is Hamilton-connected [20]. Thus, the network is (connected) non partition-able; to guarantee this, we construct HC with  $(|C_j| \geq 5)$ . For example if we construct the MON-Cluster as 4-regular, 4-connected graph, then it should have two edges-disjoint HC [21], [24].

**Definition 3 (MON-Cluster's Diameter):** The diameter of the  $2\chi$ -regular graph ( $C_j$ ) at instance of time  $\tau$  is bounded by

$$1 + 2\chi + 2\chi(2\chi - 1) + \dots + 2\chi(2\chi - 1)^{D_\tau^{(j)} - 1} = \frac{2\chi(2\chi - 1)^{D_\tau^{(j)}} - 2}{2\chi - 2} = N_\tau(2\chi, D_\tau^{(j)}) = |C_j|_\tau \tag{1}$$

This value is called the Moore bound [17], and it is known that, for  $D_\tau^{(j)} \geq 2$  and  $2\chi \geq 3$ . Where  $|C_j|_\tau$  is the estimated size of MON-Cluster  $C_j$  at instance of time  $\tau$ . From equation 1 if  $|C_j|_\tau - 1 \leq (2\chi)^{D_\tau^{(j)}}$  [18] then

$$D_\tau^{(j)} \geq \frac{\log(|C_j|_\tau - 1)}{\log 2\chi} \tag{2}$$

Thus, by estimating the size of the  $j$ -th MON-Cluster the diameter  $D_\tau^{(j)}$  can be determined.

1) MON-Cluster's Size Estimation Algorithm

Each MON-Cluster is organized as  $2\chi$ -regular graph composed of  $\chi$  independent edge-disjoint HC. In order to allow a new end-node to join a MON-Cluster, It is required to estimate the size of this MON-Cluster and thus don't allow it to expand its diameter more than  $\mu$ . The Cluster Size Estimation problem is representative of a large class of problems for collecting statistics about a

large-scale distributed system, in a decentralized manner. It has two flavors – one-shot and continuous. **MON-Cluster's Size Estimation Problem** gives a protocol that when initiated by one node, estimates the number of non-faulty nodes in the MON-Cluster containing the initiating node. This problem is impossible to solve accurately in a dynamic cluster. Notice that a cluster size estimation protocol will take non-zero time to run. This motivates us to design an algorithm for *approximate* estimation. We propose a sample one-shot estimation algorithm of the MON-Cluster's size as follows.

- i. A node  $X$  that requires to estimate the size of its MON-Cluster initiates  $Size\_Est()$  request with specific identifier  $QID$ .
- ii. Node  $X$  send that request to both its predecessor and successor in a specific  $HC$   $i$ -th.
- iii. Each node (e.g.  $Y$ ) received this query checks if this query received before or not. If  $Y$  didn't receive this query it increases the number of nodes the query has been visited by one.
- iv. Node  $Y$  sends the query to its predecessor or successor if this query was received from its successor or predecessor respectively.
- v. Repeat this process until two queries carried same  $QID$  received at a node (*meeting node*) from its successor and predecessor.
- vi. Then, *meeting node* ( $MN$ ) counts the number of nodes has been traversed by both queries.
- vii. Node  $MN$  sends the estimated MON-Cluster's size to the tutor node and the requester node.

The number of messages required to estimate the MON-Cluster's size is  $O(n)$  and  $O(n/2)$  times. Several efforts have been made to provide decentralized solutions to the cluster size estimation problem. *M. Bawa, et al* presents several mechanisms for aggregation and cluster size estimation [19]. One algorithm has an estimation query sent along a random walk within the cluster. When the estimation query hits a node for the second time, the protocol stops and the number of hops traversed so far can be used to estimate the cluster size – according to the birthday paradox, it takes  $O(\sqrt{n})$  hops for a random walk query to encounter a node a second time. Further studies concerning the network's size estimation is out the scope of this paper.

### C. $\langle \delta, \mu \rangle$ -MON Step-step Construction Scheme

This section introduces the step-step self-construction algorithm. When an end-user node  $X$  wants to participate in the multicast group then it calls  $Join\_MON()$  process that runs the following steps.

- i.  $X$  looks up for a tutor node  $t_j$  by using random walk or physical IP multicast. For example, figure 3-a shows that node  $X$  discovers the tutor node  $t_2$  of MON-Cluster  $C_2$  with diameter  $D_r^{(2)}$ . Then  $X$  sends a join request to  $t_2$ .

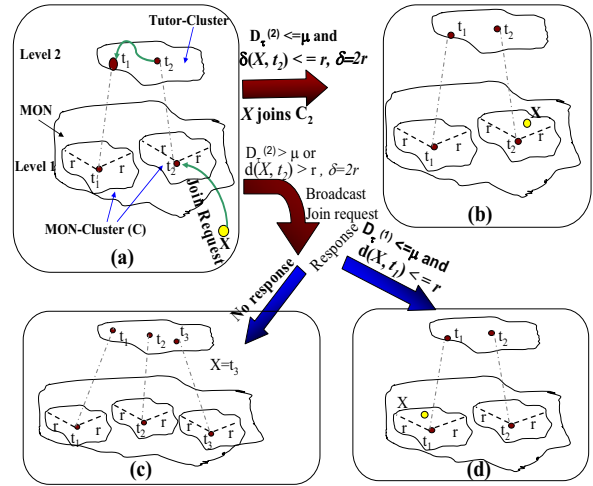


Figure 3:  $\langle \delta, \mu \rangle$ -MON Step-Step Construction

- ii. If  $[(D_r^{(j)} + 1) \leq \mu]$  and  $(d(X, t_j) \leq r; \delta = 2r)$  then node  $X$  calls  $Join\_MONCluster(C_j)$  process to join the MON-Cluster  $C_j$  as shown in figure 3-b. The  $Join\_MONCluster(C_j)$  process chooses  $\chi$  random nodes  $m_l \in C_j; (l = 1, \dots, \chi)$  and then inserts node  $X$  between each node  $l_i$  and its successor node ( $m_l \rightarrow n_s^{(i)}$ ) in the  $i$ -th  $HC$  similar to our previous work in [5].
- iii. Otherwise, if  $[d(X, t_j) > r \text{ or } ((D_r^{(j)} + 1) > \mu)]$ , the tutor node  $t_j$  broadcasts a join request message in the Tutors-Cluster and then waits *time-out*  $\alpha$  period for a reply from the tutors nodes in the Tutors-Cluster. There are two cases:
  - a) No response is received within the time-out interval  $\alpha$ , then node  $X$  creates its own new MON-Cluster and becomes a tutor node. The number of tutor nodes (MON-Clusters)  $\omega$  increases by one. For example, in figure 3-c node  $X$  becomes a tutor node  $t_3$  of the new MON-Cluster  $C_3$ . This new MON-Cluster is denoted by  $C_{\omega+1}$ .
  - b) If  $t_j$  receives multiple of replies then it selects the  $t_s$  with the smallest distance to  $X$  where  $d(X, t_s) \leq r$ ,  $[(D_r^{(s)} + 1) \leq \mu]$  and the minimum  $|C_s|_r$ . Then node  $X$  calls  $Join\_MONCluster(C_s)$  process to join the MON-Cluster  $C_s$ . For example node  $X$  joins  $C_1$  as shown in figure 3-d.

### D. Multicast Process

Every node  $X$  wants to disseminate a packet  $p$  over the  $\langle \delta, \mu \rangle$ -Multicast Overlay Network processes the following steps:

- i. Node  $X$  broadcasts the packet  $p$  to its neighbors, who in turn broadcast  $p$  to all their neighbors, except the neighbor from which it was received.
- ii. When a tutor receives the packet  $p$ , then it broadcasts this packet to its neighbors in the MON-Cluster.
- iii. In addition, each tutor node in turn broadcasts  $p$  to its neighbors in the Tutors-Cluster.

iv. This application level-multicast process goes on till the message packet  $p$  reaches all nodes in  $MON$ . In the multicast process, each node/tutor autonomously monitors the received packets to forward only one copy of the duplicated received ones.

*E.  $\langle \delta, \mu \rangle$ -MON Fault-tolerance*

**a) End Node Leave:** When node  $X$  decides to leave the  $MON$ -Cluster, it notifies its neighbors and then creates an edge between its successor and predecessor end nodes in each clockwise  $i$ -th  $HC$  as follows.

$$\begin{aligned} right\_neighbor^{(i)} &= X \rightarrow n_p^{(i)}; \\ left\_neighbor^{(i)} &= X \rightarrow n_s^{(i)}; \\ right\_neighbor^{(i)} \rightarrow n_s^{(i)} &= left\_neighbor^{(i)}; \\ left\_neighbor^{(i)} \rightarrow n_p^{(i)} &= right\_neighbor^{(i)}; \end{aligned}$$

Obviously, the leave process requires only local changes in the network with  $O(\chi)$  messages [5].

**b) End-Node Failure:** It is also required to consider the difficult case of end node failure. For fault-tolerance, we assumed that each end node knows the predecessor of its predecessor node and the successor of its successor node in each cycle. The node failure is detected locally as follows. The neighboring nodes of the failed node  $X$ , periodically exchange keep-alive messages with node  $X$ . If node  $X$  is unresponsive for a period  $\gamma$ , it is presumed it failed. Thus, each neighbor node sets

$$\begin{aligned} right\_neighbor^{(i)} \rightarrow n_s^{(i)} &= \\ &right\_neighbor^{(i)} \rightarrow n_s^{(i)} \rightarrow n_s^{(i)} \\ left\_neighbor^{(i)} \rightarrow n_p^{(i)} &= \\ &left\_neighbor^{(i)} \rightarrow n_p^{(i)} \rightarrow n_p^{(i)} \end{aligned}$$

for each  $i$ -th  $HC$ . This fault-tolerance process connects two nodes around the failed node in the same cycle and sets the predecessor and successor nodes to its predecessor and successor. It maintains the  $HC$ . This technique scales well: fault detection is done by exchanging messages among small number of nodes, and recovery from faults is local; only a small number of nodes ( $2\chi$ ) is involved. In addition, this technique maintains the  $MON$ -Cluster composed of disjoint  $HC$ . If a Hamilton path connects every two nodes of  $G$  then  $G$  is *Hamilton-connected* [20]. Thus, the network is (connected) non partition-able; to guarantee this, we constructed  $HC$  with  $\left( \left| C_j \right|_{\tau} \geq 5 \right)$ . For example if we construct the  $MON$ -Cluster as 4-regular, 4-connected graph, then it should have two edges-disjoint  $HC$  [21].

**c. Tutor node Leave:** When a tutor node wants to leave. It selects one of its neighbors randomly to act instead of it after his leave.

**d. Tutor node Failure:** Periodically, the neighbors of the tutor node  $t_j$  exchange keep-alive messages with  $t_j$  to detect its failure. When  $t_j$  failed then one from its neighbors becomes a tutor node. Then this new tutor node sends join request to the *Tutors-Cluster*. In addition, in the *Tutors-Cluster*, the neighbors of the failed  $t_j$  detect its failure and then remove their links with it and wait for

new tutor node to join instead. A small number of nodes ( $3\chi$ ) is involved in the Tutor node fault tolerant process. The details discussion and evaluation of the proposed fault-tolerance are out the scope of this paper. For more details please refer to [5].

IV. PERFORMANCE EVALUATION

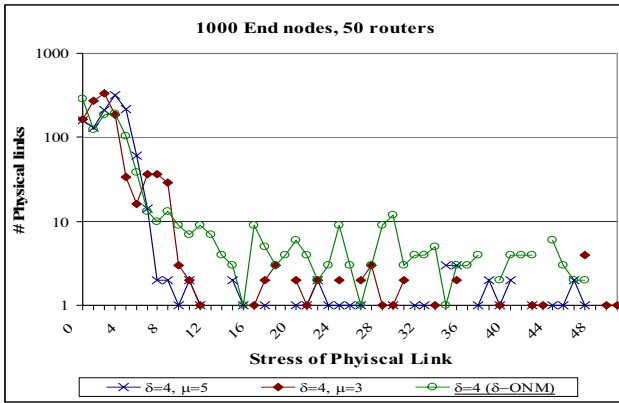
This section demonstrates the proposed application level multicast over  $\langle \lambda, \mu \rangle$ -MON. To evaluate the effectiveness of the proposed  $\langle \lambda, \mu \rangle$ -MON step-step construction scheme of  $MON$ , we first compare it with  $\delta$ -MON. The  $\delta$ -MON is an overlay network that is constructed with topology awareness. It organized the  $MON$  into  $MON$  clusters, where the physical distance between any nodes in each cluster is less than or equal to  $\delta$ . The  $\delta$ -MON expands the  $MON$ -Cluster without size limit. In addition, we use the following metrics.

**Mean Maximum Communication Delay (MMCD).** It measures mean of the maximum communication delay required to broadcast a message from any end-node to all end-nodes in  $MON$ . The simulation picks a random end-node that initiates a message and then broadcasts it to all nodes in  $MON$ . Then the simulation determines the required *Maximum Communication Delay (MCD)* to broadcast that message into  $MON$ . The simulation repeats this process by selecting different random end-nodes until all nodes broadcast the same message. Finally, it determines the *MMCD*.

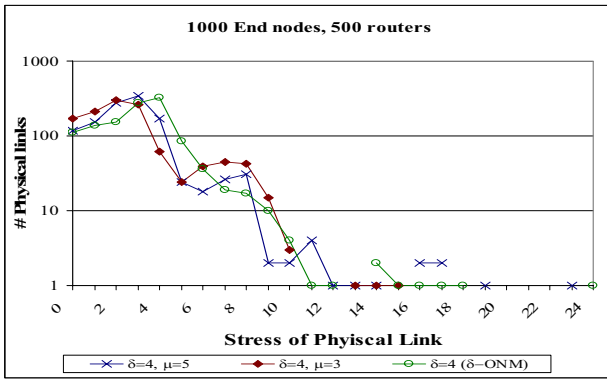
**Physical Link Stress,** a measure of how the proposed  $\langle \delta, \mu \rangle$ -MON constructing scheme for  $MON$  that is effective in distributing network load across different physical links. It refers to the number of identical copies of a packet carried by a physical link for service discovery. The simulation picks random node that send a packet and then measures the worst stress value of all physical links.

A. Simulation Setup

The simulation consists of 50-500 backbone routers linked by core links over the underlying topology *transit-stub* model. The *Georgia Tech* [22] random graph generator is used to create that network model. Random link delay of 4-12ms was assigned to each core link. The *end nodes* were randomly assigned to routers in the core with uniform probability. Each end-node was directly attached by a LAN link to its assigned router. The delay of each LAN link was set to be 1ms. The *transit-stub* network model consists of three stub domains per transit node, with no extra transit-stub or stub-stub edges. The edge probability between each pair of nodes within each stub domain is 0.42, 0.6, and 1.0 respectively. The simulation ran with different number of end-nodes  $N$  that is ranged from 100 - 1000.



(a)



(b)

Figure 4: Physical Links Stress

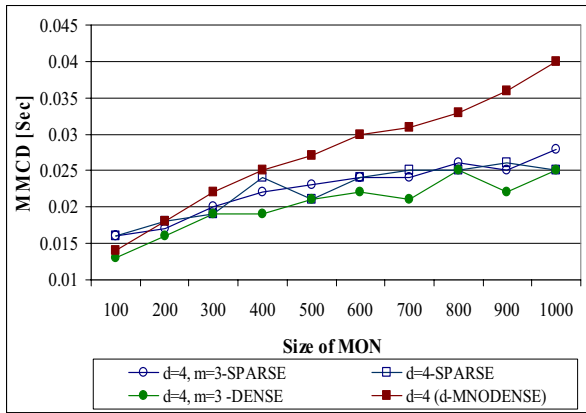


Figure 5: Multicast Communication Delay

B. Simulation Results

Figures 4-a, 4-b plot the number of physical links suffering from a particular stress level on the Y-axis, against the range of stress levels on the X-axis. Figure 4-a proves that the multicast’s traffic overhead induced over  $\langle \delta, \mu \rangle$ -MON is less than the traffics overhead induced over  $\delta$ -MON by 37% when the number of end nodes  $N$  is 1000, the number of routers  $R$  is 50,  $\delta=4$  and  $\mu=3$  while it is 13% when  $R=500$  as shown in figure 4-b. Moreover, the  $\langle \delta, \mu \rangle$ -MON improves the links stress by 9% when  $R=50$  and  $\mu=3$  than when  $\mu=5$ , similarly when  $R=500$  the link stress is improved by 5% when  $\mu=3$  than when  $\mu=5$ . This proves that the proposed  $\langle \delta, \mu \rangle$ -MON construction scheme reduces the links stress more in dense network

topologies. We argue that to the  $\langle \delta, \mu \rangle$ -MON proposal limits the expansion of the MON-cluster.

Figure 5 plots the variations of the MMCD along with the MON’s size over dense and sparse transit-stub physical network with different sizes 50, and 500 routers respectively. Figure 4 shows that the communication delay over  $\langle \delta, \mu \rangle$ -MON is less than the communication delay over  $\delta$ -MON by 3% and 0.8% in dense and sparse networks respectively. Moreover it shows that for small MON’s size is not effective but it shows improvement of the communication delay by 37% when  $N=1000$  in dense network and about 1% in sparse network. This proves that the  $\langle \delta, \mu \rangle$ -MON is efficient for dense networks. Finally, the simulation results mentioned before give an insight that the proposed  $\langle \delta, \mu \rangle$ -MON construction scheme constructs MON that enables efficient and scalable multicast service.

REFERENCES

- [1] I. Clarke, Et al. “Freenet: A distributed anonymous information storage and retrieval system,” Proc. Of Workshop on Design Issues in Anonymity and Unobservability, 2001.
- [2] J. Kubiawicz, Et al. “Oceanstore: An Architecture for global-scale persistent storage,” Proc. Of ACM ASPLOS, 2000.
- [3] M. Ripeanu, “Peer-to-Peer architecture case study: Gnutella network,” Proc. Of 1<sup>st</sup> conference on P2P Computing, August 2001.
- [4] D. Zeinalipour, “Information retrieval techniques for Peer-to-peer Networks,” IEEE Computing in Science and Engineering magazine, Vol. 6 No. 4 pp. 20-26, July/August, 2004.
- [5] K. Ragab, Et. al. “Autonomous Decentralized Community Communication for Information Dissemination,” IEEE Internet Computing Magazine, Vol. 8 No. 3, pp. 29-36, May/June 2004.
- [6] D. Pendarakis, “ALMI: an application level multicast infrastructure”, In Proc. of 3rd Symp. USITS, March 2001
- [7] P. Francis, “YOID: Extending the Multicast Architecture,” 1999, White paper <http://www.aciri.org/yoid>.
- [8] Y. H. Chu, Et al., “A case for end system multicast,” in Proc. Of ACM Sigmetrics, June 2000.
- [9] Y. Chawathe., “Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service,” Ph.D Thesis, Univ. of California, Berkeley, Dec. 2000.
- [10] S. Ratnasamy, Et al., “A Scalable Content- Addressable Network,” Proc. Of SIGCOMM’01, California, USA.
- [11] K. Ragab, K. Mori, “ACIS-Hierarchy: Enhancing Community Communication Delay for Large-Scale Information Systems,” IEICE Trans. on Communication, Vol. E87-B, No. 7, pp. , July 2004
- [12] K. Ragab “Autonomous Decentralized Community Communication Technology for Assuring Information Dissemination,” Ph.D. Thesis Tokyo Institute of Technology, 2004.
- [13] Defering, S. E. “Multicast Routing in a Datagram Internetwork,” PhD thesis, Stanford University, Dec. 1991.
- [14] S. Q. Zhuang, Et al., “Bayeux: An Architecture for Scalable and Fault tolerant Wide-area Data

- Dissemination,” In Proc. NOSSDAV Workshop, June 2001.
- [15] B. Awerbuch and D. Peleg, “Sparse Partitions,” IEEE Proc. Of Symposium on Foundation of Computer Science, 1990.
- [16] B. Jackson and H. Li, “Hamilton Cycles in 2-Connected  $k$ -regular Bipartite Graphs,” *Comb. Theory Series A*, 44 (2): 177-186, 1998.
- [17] N. Biggs, “Algebraic Graph Theory”, Cambridge Math. Library ISBN 0-521-45897-8 pbk, (1974, 1993 (2nd edition)).
- [18] A. H. Dekker, B. D. Colbert, “Network robustness and graph topology,” *ACM Proc. 27<sup>th</sup> Int. Conf. on Australian Computer Science*, pp 359-368, 2004.
- [19] M. Bawa, H. Garcia-Molina, A. Gionis and R. Motwani, “Estimating aggregates on a peer-to-peer network,” Technical Report, Dept. of Computer Science, Stanford University, 2003.
- [20] J. A. Bondy and U.S. R. Murty, “Graph Theory with Applications,” Macmilliam Press Ltd. 1976.
- [21] Tutte W.T., “A Theorem on Planner graphs,” *Trans/Amer. Math Soc.* 82 (1956), 99-116
- [22] E. W. Zegura, “How to model an Internetwork,” *IEEE-Proc. INFOCOM*, 1996, San Francisco.
- [23] S. Ratnasamy, Et. al., “Application-level multicast using content-addressable networks,” *Proc of NGC*, 2001.
- [24] Granbaum B. Et al., “Pairs of Edge-disjoint Hamiltonian Circuit” *Journal of Aequationes Mathematicae*, vol. 4, No. 1-2, 1976.



**K. Ragab** is an assistant professor at Department of Computer Science, College of Computer and Information Technology, King Faisal University, Saudi Arabia. Moreover, he is on leave assistant professor of Computer Science at Department of Mathematic, Computer Science division, Ain Shams University. He joined Department of Computer Science, Tokyo University in 2005 as postdoctoral

position. He was born in 1968 and received his B.Sc., M.Sc. degrees in Computer Science from Ain Shams University, Cairo, Egypt in 1990, 1999, respectively and Ph.D. degree in Computer Science from Tokyo Institute of Technology in 2004. He has worked in Ain Shams University, Cairo Egypt in 1990-1999 as assistant lecturer. He has worked as research scientist in Computer Science Dept., Technical University of Chemnitz, Germany in 1999-2001. His research interests include autonomous decentralized systems, Peer-to-Peer Systems, Overlay Networks, Web-services and application-level multicast.



**A. Yonezawa** is professor of computer science at Department of Computer Science, University of Tokyo. He was born in 1947, and received his B.E. and M.E. degrees from University of Tokyo in 1970, and 1972, and Ph.D. in Computer Science from the Massachusetts Institute of Technology in 1977. His current major research interests are in the areas of concurrent/parallel computation models, programming languages, object-oriented computing, and distributed computing. He is the designer of an object-oriented concurrent language ABCL/1 and currently leading the research project on object-based concurrency and distribution. He is one of the pioneers of object-based/oriented concurrent computing: he not only designed concurrent object languages. He was a member of the Scientific Advisory Board of German National Research Institute of Computer Science (GMD), and is currently the president of Japanese Society of Software Science and Technology and a member of the Evaluation and Promotion Committee of Japanese MITI's Real World Computing Program. He was also appointed by Prime Minister to a member of the Reformation and Deregulation Committee and the chairman of its Education Subcommittee. Since April 2006, he has been the director of the Information Technology Center, University of Tokyo.